Final Database Project

Kai Wong

CMPT308N112

Kai Wong
Schwartz
CMPT308N112
5 Dec 2016

Table of Contents

Database Model Description

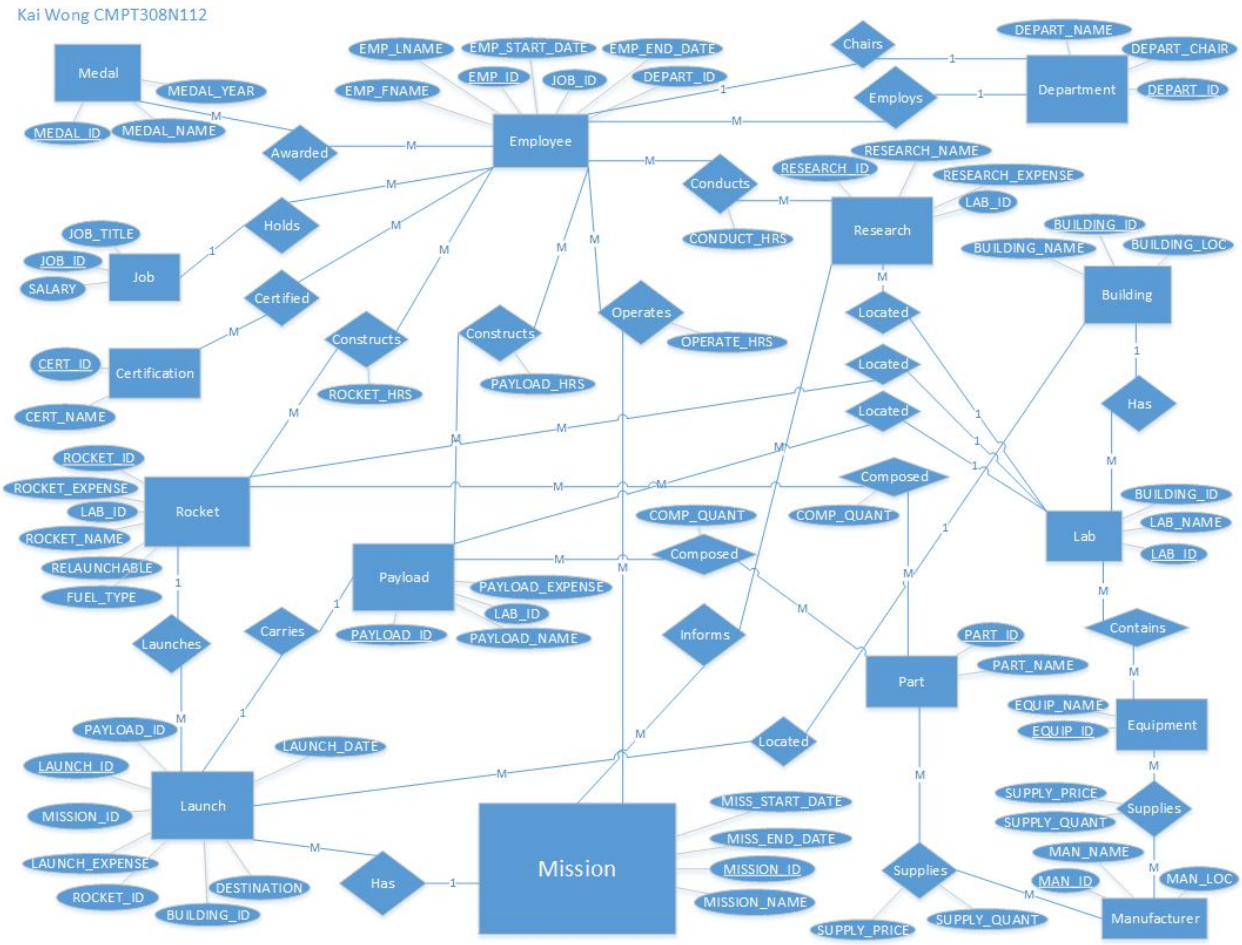
ASAN, a small space company, needs a system to effectively manage its research projects, construction projects, missions, launches, parts, equipment, funding, and staff. By using this system, ASAN will see greater efficiency in the work that the agency carries out. The system must be designed to satisfy the following requirements:

1. ASAN has employees. Each employee has a unique employee number, a first name, a last name, a job, a department they work for, the date they started working for ASAN, and the date they finish/will finish working for ASAN. An employee can be awarded medals, can be certified with different certifications, can work on rocket and payload constructions, can operate missions, and can conduct research.
2. ASAN has multiple departments. Each department has a unique department number, a name, and an employee who chairs the department. An employee can only chair one department.
3. ASAN gives medals to its employees. Each medal has a unique medal number, a medal year, and a medal name. Multiple employees can receive the same award.
4. ASAN has jobs. Each job has a unique job number, a job title, and an hourly salary. Multiple employees can hold the same job.
5. ASAN's employees are certified in different areas. Each certification has a unique certification number, and a name. Multiple employees can hold the same certification, and can hold multiple certifications.
6. ASAN's employees can construct rockets, and they each construct rockets for a number of hours. Each rocket has a unique rocket number, an expense to build the rocket, the lab it is being built in, a name, whether it is relaunchable or not, and what fuel type it uses. An employee can be working on multiple rockets at the same time, and a rocket has multiple employees constructing it. Since a rocket can be relaunchable, a rocket can be launched multiple times carrying different payloads.
7. ASAN's employees can construct payloads, and they each construct payloads for a number of hours. Each payload has a unique payload number, an expense to build the payload, the lab it is being built in, and a name. An employee can be constructing multiple payloads at the same time, and a payload has multiple employees constructing it. A payload can only be launched once.
8. ASAN's employees can perform research, and they each perform research for a number of hours. Each research project performed has a research number, a name, a research expense, and the lab it is being performed in. An employee can be performing multiple research projects at the same time, and a research

project can have multiple employees conducting research. Research projects also supply missions with information to work off of.

9. ASAN has buildings located all across the country in different states. Each building has a unique number, a name, and the location by state it is in. Each building has one or more labs. A lab has a unique number, a name, and the building number it is located in. Each lab contains equipment, and labs can be used to constructs rockets, payloads, or can be used to conduct research.
10. ASAN has equipment that is stored in labs. Each piece of equipment has a unique number and a name, and has a quantity. The same type of equipment can be stored in multiple labs.
11. ASAN has manufacturers that supply them their equipment and parts. Each manufacturer has a unique number, a name, and a location by state that they are in. ASAN is supplied parts and equipment by price and quantity.
12. ASAN has parts they use to construct their payloads and rockets. Each part has a unique number and a name. A part can be supplied by multiple manufacturers, and a part can be used in multiple payloads and rockets.
13. ASAN has launches for their rockets and payloads. Each launch has a unique number, the mission number it is associated with, the number of the payload it is carrying, the expense for the launch, the rocket number, the building where the launch is located at, the destination of the launch, and the date of the launch. Each launch can only have one rocket and one payload.
14. ASAN has missions. Each mission has a unique number, a name, a start date and an end date. A mission can have multiple launches, can be supported by multiple research projects, and are operated by employees. Employees operate missions for a certain number of hours.

E/R Diagram (Chen)



Follow this link (http://imgur.com/a/656hy) to see a bigger version, or see iLearn submission.

Building Table

```
CREATE TABLE Building(
        BUILDING_ID INT PRIMARY KEY,
        BUILDING_NAME VARCHAR(255) NOT NULL,
        BUILDING_LOC VARCHAR(255) NOT NULL);
```

This table creates the Building entity, with BUILDING_ID, BUILDING_NAME, and BUILDING_LOC as its attributes. BUILDING_ID is the primary key and is an integer, BUILDING_NAME is a string of the name of the building, and BUILDING_LOC is a string of the location of the building by state.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (BUILDING_NAME and BUILDING_LOC are functionally dependent on BUILDING_ID).

Certification Table

```
CREATE TABLE Certification(
        CERT_ID INT PRIMARY KEY,
        CERT_NAME VARCHAR(255) NOT NULL);
```

This table creates the Certification entity. It has CERT_ID and CERT_NAME as its attributes. CERT_ID is the primary key and is an integer, and CERT_NAME is the string for the name of the certification.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (CERT_NAME is functionally dependent on CERT_ID).

Department Table

```
CREATE TABLE Department(
        DEPART_ID INT PRIMARY KEY,
        DEPART_NAME VARCHAR(255) NOT NULL,
        DEPART_CHAIR INT NOT NULL,
        FOREIGN KEY (DEPART_CHAIR) REFERENCES EMPLOYEE (EMP_ID));
```

This creates the Department entity. It has DEPART_ID, DEPART_NAME, and DEPART_CHAIR as its attributes. DEPART_ID is the primary key and is an integer, DEPART_NAME is a string of the department name, and DEPART_CHAIR is an integer of the employee who chairs the department. DEPART_CHAIR is a foreign key and references an employee number in the EMPLOYEE table.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (DEPART_NAME and DEPART_CHAIR are functionally dependent on DEPART_ID).

Emp_Award Table

```
CREATE TABLE Emp_Award(
        MEDAL_ID INT NOT NULL,
        EMP_ID INT NOT NULL,
        PRIMARY KEY (MEDAL_ID, EMP_ID),
        FOREIGN KEY (MEDAL_ID) REFERENCES MEDAL (MEDAL_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID));
```

This creates the Emp_Award entity. It has MEDAL_ID and EMP_ID as its attributes. This table was created out of the many-to-many relationship between EMPLOYEE and MEDAL. The primary key is MEDAL_ID and EMP_ID, which are both integers, and they represent the medal and the employee respectively. MEDAL_ID is also a foreign key and references the medal number in the MEDAL table, and EMP_ID is also a foreign key and references the employee number in the EMPLOYEE table.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (there are no non-prime attributes to functionally depend on part of the composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there are no non-prime attributes to be functionally dependent on each other).

Emp_Cert Table

```
CREATE TABLE Emp_Cert(
        CERT_ID INT NOT NULL,
        EMP_ID INT NOT NULL,
        PRIMARY KEY (CERT_ID, EMP_ID),
        FOREIGN KEY (CERT_ID) REFERENCES CERTIFICATION (CERT_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID));
```

This creates the Emp_Cert entity. This table, created out of the many-to-many relationship between EMPLOYEE and CERTIFICATION, has CERT_ID and EMP_ID as its attributes. CERT_ID and EMP_ID make up the primary key, are both integers, and represent the certification and the employee respectively, CERT_ID is a foreign key and references the certification number in CERTIFICATION, and EMP_ID is a foreign key and references the employee number in EMPLOYEE.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (there are no non-prime attributes to functionally depend on part of the composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there are no non-prime attributes to functionally depend on each other).

Emp_Conducts Table

```
CREATE TABLE Emp_Conducts(
        EMP_ID INT NOT NULL,
        RESEARCH_ID INT NOT NULL,
        CONDUCT_HRS INT NOT NULL,
        PRIMARY KEY (EMP_ID, RESEARCH_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID),
        FOREIGN KEY (RESEARCH_ID) REFERENCES RESEARCH (RESEARCH_ID));
```

This creates the Emp_Conducts entity. This table was created out of the many-to-many relationship between RESEARCH and EMPLOYEE, and has EMP_ID, RESEARCH_ID, and CONDUCT_HRS as its attributes. EMP_ID and RESEARCH_ID make up the primary key, are integers, and represent the employee and the research project respectively. EMP_ID is a foreign key and references the employee number in EMPLOYEE, and RESEARCH_ID is a foreign key and references the research number in RESEARCH. CONDUCT_HRS is the integer number of hours an employee has conducted research.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute CONDUCT_HRS functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Emp_Constructs_Payload Table

```
CREATE TABLE Emp_Constructs_Payload(
        EMP_ID INT NOT NULL,
        PAYLOAD_ID INT NOT NULL,
        PAYLOAD_HRS INT NOT NULL,
        PRIMARY KEY (EMP_ID, PAYLOAD_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID),
        FOREIGN KEY (PAYLOAD_ID) REFERENCES PAYLOAD (PAYLOAD_ID));
```

This creates the Emp_Constructs_Payload entity. This table was created from the many-to-many relationship between EMPLOYEE and PAYLOAD, and has EMP_ID, PAYLOAD_ID, and PAYLOAD_HRS as its attributes. EMP_ID and PAYLOAD_ID make up its primary key, are integers, and represent the employee and the payload respectively, while EMP_ID is a foreign key that references the employee number in EMPLOYEE, and PAYLOAD_ID is a foreign key that references the payload number in PAYLOAD. PAYLOAD_HRS is the integer number of hours an employee has constructed a payload.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute PAYLOAD_HRS functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Emp_Constructs_Rocket Table

```
CREATE TABLE Emp_Constructs_Rocket(
        EMP_ID INT NOT NULL,
        ROCKET_ID INT NOT NULL,
        ROCKET_HRS INT NOT NULL,
        PRIMARY KEY (EMP_ID, PAYLOAD_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID),
        FOREIGN KEY (ROCKET_ID) REFERENCES ROCKET (ROCKET_ID));
```

This creates the Emp_Constructs_Rocket entity. This table was created out of the many-to-many relationship between EMPLOYEE and ROCKET, and EMP_ID, ROCKET_ID, and ROCKET_HRS make up its attributes. EMP_ID and PAYLOAD_ID make up its primary key, are integers, and represent the employee and the payload respectively. EMP_ID is a foreign key that references the employee number in EMPLOYEE, and ROCKET_ID is a foreign key that references the rocket number in ROCKET. ROCKET_HRS is the integer number of hours that an employee has constructed a rocket.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute ROCKET_HRS functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Emp_Operates Table

```
CREATE TABLE Emp_Operates(
        EMP_ID INT NOT NULL,
        MISSION_ID INT NOT NULL,
        OPERATE_HRS INT NOT NULL,
        PRIMARY KEY (EMP_ID, MISSION_ID),
        FOREIGN KEY (EMP_ID) REFERENCES EMPLOYEE (EMP_ID),
        FOREIGN KEY (MISSION_ID) REFERENCES MISSION (MISSION_ID));
```

This creates the Emp_Operates entity. This table was created from the many-to-many relationship between MISSION and EMPLOYEE. Its attributes include EMP_ID, MISSION_ID, and OPERATE_HRS. The primary key is made up of EMP_ID and MISSION_ID, are integers, and represent the employee and the mission respectively. EMP_ID is a foreign key to the employee number in EMPLOYEE, and MISSION_ID is a foreign key to the mission number in MISSION. OPERATE_HRS is the integer number of hours that an employee has operated a mission.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute OPERATE_HRS functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Employee Table

```
CREATE TABLE Employee(
        EMP_ID INT PRIMARY KEY,
        EMP_FNAME VARCHAR(255) NOT NULL,
        EMP_LNAME VARCHAR(255) NOT NULL,
        JOB_ID INT NOT NULL,
        DEPART_ID INT NOT NULL,
        EMP_START_DATE DATE NOT NULL,
        EMP_END_DATE DATE NOT NULL,
        FOREIGN KEY (JOB_ID) REFERENCES JOB (JOB_ID),
        FOREIGN KEY (DEPART_ID) REFERENCES DEPARTMENT (DEPART_ID));
```

This creates the Employee entity. Its attributes are EMP_ID (an integer), EMP_FNAME (a string), EMP_LNAME (a string), JOB_ID (an integer), DEPART_ID (an integer), EMP_START_DATE (a date), and EMP_END_DATE (a date). EMP_ID is the primary key for this entity. JOB_ID is a foreign key that references the job number in JOB, and DEPART_ID is a foreign key that references the department number in DEPARTMENT. EMP_FNAME and EMP_LNAME is the employee's full name, JOB_ID is the job that the employee has, DEPART_ID is the department the employee works for, and EMP_START_DATE and EMP_END_DATE are the start and end dates for the employee.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (EMP_FNAME, EMP_LNAME, JOB_ID, DEPART_ID, EMP_START_DATE, and EMP_END_DATE all functionally depend on EMP_ID).

Equipment Table

```
CREATE TABLE Equipment(
        EQUIP_ID INT PRIMARY KEY,
        EQUIP_NAME VARCHAR(255) NOT NULL);
```

This creates the Equipment entity. Its attributes include EQUIP_ID and EQUIP_NAME. EQUIP_ID is an integer and the primary key, and EQUIP_NAME is a string of the name of the equipment.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (EQUIP_NAME is functionally dependent on EQUIP_ID).

Informs Table

```
CREATE TABLE Informs(
        RESEARCH_ID INT NOT NULL,
        MISSION_ID INT NOT NULL,
        PRIMARY KEY (RESEARCH_ID, MISSION_ID),
        FOREIGN KEY (RESEARCH_ID) REFERENCES RESEARCH (RESEARCH_ID),
        FOREIGN KEY (MISSION_ID) REFERENCES MISSION (MISSION_ID));
```

This creates the Informs entity. This table was created due to the many-to-many relationship between RESEARCH and MISSION. Its attributes include the integers RESEARCH_ID and MISSION_ID, and they make up the primary key. RESEARCH_ID represents the research project, and MISSION_ID represents the mission. RESEARCH_ID is a foreign key that references the research number in RESEARCH, and MISSION_ID is a foreign key that references the mission number in MISSION.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (there are no non-prime attributes to functionally depend on part of the composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there are no non-prime attributes to functionally depend on each other).

Job Table

```
CREATE TABLE Job(
        JOB_ID INT PRIMARY KEY,
        JOB_TITLE VARCHAR(255) NOT NULL,
        SALARY INT NOT NULL);
```

This creates the Job entity. Its attributes include JOB_ID (an integer and the primary key), JOB_TITLE (a string of the name of the job), and SALARY (an integer of the hourly rate).

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (JOB_TITLE AND SALARY are functionally dependent on JOB_ID).

Lab Table

```
CREATE TABLE Lab(
        LAB_ID INT PRIMARY KEY,
        LAB_NAME VARCHAR(255) NOT NULL,
        BUILDING_ID INT NOT NULL,
        FOREIGN KEY (BUILDING_ID) REFERENCES BUILDING (BUILDING_ID));
```

This creates the Lab entity. It is made up of LAB_ID (an integer and the primary key), LAB_NAME (a string of the name of the lab), and BUILDING_ID (an integer that represents the building the lab is in). BUILDING_ID is a foreign key that references the building number in BUILDING.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (LAB_NAME AND BUILDING_ID are functionally dependent on LAB_ID).

Lab_Contains Table

```
CREATE TABLE Lab_Contains(
        LAB_ID INT NOT NULL,
        EQUIP_ID INT NOT NULL,
        EQUIP_QUANT INT NOT NULL,
        PRIMARY KEY (LAB_ID, EQUIP_ID),
        FOREIGN KEY (LAB_ID) REFERENCES LAB (LAB_ID),
        FOREIGN KEY (EQUIP_ID) REFERENCES EQUIPMENT (EQUIP_ID));
```

This creates the Lab_Contains entity. This table was created from the many-to-many relationship between LAB and EQUIPMENT. Its attributes consist of LAB_ID, EQUIP_ID, and EQUIP_QUANT. The primary key consists of LAB_ID and EQUIP_ID, both integers that represent the lab and the equipment respectively, and EQUIP_QUANT is the integer quantity of equipment in that lab. LAB_ID is a foreign key that references the lab number in LAB, and EQUIP_ID is a foreign key that references the equipment number in EQUIPMENT.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute EQUIP_QUANT functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Launch Table

```
CREATE TABLE Launch(
        LAUNCH_ID INT PRIMARY KEY,
        MISSION_ID INT NOT NULL ,
        PAYLOAD_ID INT NOT NULL,
        LAUNCH_DATE DATE NOT NULL,
        DESTINATION VARCHAR(255) NOT NULL,
        ROCKET_ID INT NOT NULL,
        BUILDING_ID INT NOT NULL,
        LAUNCH_EXPENSE INT NOT NULL,
        FOREIGN KEY (MISSION_ID) REFERENCES MISSION (MISSION_ID),
        FOREIGN KEY (PAYLOAD_ID) REFERENCES PAYLOAD (PAYLOAD_ID),
        FOREIGN KEY (BUILDING_ID) REFERENCES BUILDING (BUILDING_ID),
        FOREIGN KEY (ROCKET_ID) REFERENCES ROCKET (ROCKET_ID));
```

This creates the Launch entity. Its attributes include LAUNCH_ID (an integer and the primary key), MISSION_ID (an integer representing the mission the launch belongs to), PAYLOAD_ID (an integer representing the payload being launched), LAUNCH_DATE (a date representing day of launch), DESTINATION (a string representing where the payload of the launch is headed to), ROCKET_ID (an integer representing the rocket used for the launch), BUILDING_ID (an integer representing what building the launch is taking place at), and LAUNCH_EXPENSE (an integer representing the cost for the launch). MISSION_ID is a foreign key that references the mission number in MISSION, PAYLOAD_ID is a foreign key that references the payload number in PAYLOAD, BUILDING_ID is a foreign key that references the building number in BUILDING, and ROCKET_ID is a foreign key that references the rocket number in ROCKET.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (MISSION_ID, PAYLOAD_ID, LAUNCH_DATE, DESTINATION, ROCKET_ID, BUILDING_ID, and LAUNCH_EXPENSE are all functionally dependent on LAUNCH_ID).

Manufacturer Table

```
CREATE TABLE Manufacturer(
        MAN_ID INT PRIMARY KEY,
        MAN_NAME VARCHAR(255) NOT NULL,
        MAN_LOC VARCHAR(255) NOT NULL);
```

This creates the Manufacturer entity. Its attributes consist of MAN_ID (an integer and the primary key), MAN_NAME (a string representing the name of the manufacturer), and MAN_LOC (a string representing the location of the manufacturer by state).

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (MAN_NAME AND MAN_LOC are functionally dependent on MAN_ID).

Medal Table

```
CREATE TABLE Medal(
        MEDAL_ID INT PRIMARY KEY,
        MEDAL_YEAR INT NOT NULL,
        MEDAL_NAME VARCHAR(255) NOT NULL;
```

This creates the Medal entity. Its attributes consist of MEDAL_ID (an integer and the primary key), MEDAL_YEAR (an integer representing the year of the medal), and MEDAL_NAME (a string representing the name of the medal)

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (MEDAL_YEAR and MEDAL_NAME are functionally dependent on MEDAL_ID).

Mission Table

```
CREATE TABLE Mission(
        MISSION_ID INT PRIMARY KEY,
        MISSION_NAME VARCHAR(255) NOT NULL,
        MISS_START_DATE DATE NOT NULL,
        MISS_END_DATE DATE DATE NOT NULL);
```

This creates the Mission entity. Its attributes consist of MISSION_ID (an integer and the primary key), MISSION_NAME (a string representing the name of the mission), MISS_START_DATE (a date representing the start date of the mission) and MISS_END_DATE (a date representing the end date of the mission).

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (MISSION_NAME, MISS_START_DATE, and MISS_END_DATE are all functionally dependent on MISSION_ID).

Part Table

```
CREATE TABLE Part(
        PART_ID INT PRIMARY KEY,
        PART_NAME VARCHAR(255) NOT NULL);
```

This creates the Part entity. Its attributes are made up of PART_ID (an integer and the primary key), and PART_NAME (a string representing the name of the part).

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (PART_NAME is functionally dependent on PART_ID).

Payload Table

```
CREATE TABLE Payload(
        PAYLOAD_ID INT PRIMARY KEY,
        PAYLOAD_NAME VARCHAR(255) NOT NULL,
        LAB_ID INT NOT NULL,
        PAYLOAD_EXPENSE INT NOT NULL,
        FOREIGN KEY (LAB_ID) REFERENCES LAB (LAB_ID));
```

This creates the Payload entity. Its attributes consist of PAYLOAD_ID (which is the primary key and an integer), PAYLOAD_NAME (a string representing the name of the payload), LAB_ID (an integer representing what lab the payload is being built in), and PAYLOAD_EXPENSE (an integer representing the cost to building the payload). LAB_ID is a foreign key that references the lab number in LAB.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (PAYLOAD_NAME, LAB_ID, and PAYLOAD_EXPENSE are all functionally dependent on PAYLOAD_ID).

Payload_Composed_Of Table

```
CREATE TABLE Payload_Composed_Of(
        PAYLOAD_ID INT NOT NULL,
        PART_ID INT NOT NULL,
        COMP_QUANT INT NOT NULL,
        PRIMARY KEY (PAYLOAD_ID, PART_ID),
        FOREIGN KEY (PAYLOAD_ID) REFERENCES PAYLOAD (PAYLOAD_ID),
        FOREIGN KEY (PART_ID) REFERENCES PART (PART_ID));
```

This creates the Payload_Composed_Of entity. This table was created from the many-to-many relationship between PAYLOAD and PART. Its attributes consist of PAYLOAD_ID (an integer representing the payload), PART_ID (an integer representing the part), and COMP_QUANT (an integer representing the number of each part that composes the payload). The primary key is made up of PAYLOAD_ID and PART_ID, PAYLOAD_ID is a foreign key that references the payload number in PAYLOAD, and PART_ID is a foreign key that references the part number in PART.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute COMP_QUANT functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Research Table

```
CREATE TABLE Research(
        RESEARCH_ID INT PRIMARY KEY,
        RESEARCH_NAME VARCHAR(255) NOT NULL,
        RESEARCH_EXPENSE INT NOT NULL,
        LAB_ID INT NOT NULL,
        FOREIGN KEY (LAB_ID) REFERENCES LAB (LAB_ID));
```

This creates the Research entity. Its attributes include RESEARCH_ID (an integer and the primary key), RESEARCH_NAME (a string representing the name of the research project), RESEARCH_EXPENSE (an integer representing the cost of the research project), and LAB_ID (an integer representing the lab the research is taking place in). LAB_ID is a foreign key that references a lab number in LAB.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (this is given because the primary key is not composite).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (RESEARCH_NAME, RESEARCH_EXPENSE, and LAB_ID are functionally dependent on RESEARCH_ID).

Rocket Table

```
CREATE TABLE Rocket(
        ROCKET_ID INT PRIMARY KEY,
        ROCKET_NAME VARCHAR(255) NOT NULL,
        ROCKET_EXPENSE INT NOT NULL,
        FUEL_TYPE VARCHAR(255) NOT NULL,
        RELAUNCHABLE CHAR(1) NOT NULL,
        LAB_ID INT NOT NULL,
        FOREIGN KEY (LAB_ID) REFERENCES LAB (LAB_ID));
```

This creates the Rocket entity. Its attributes include ROCKET_ID (an integer and the primary key), ROCKET_NAME (a string representing the name of the rocket), ROCKET_EXPENSE (an integer representing the cost of the rocket), FUEL_TYPE (a string representing the type of fuel the rocket consumes), RELAUNCHABLE (a single character value, either Y or N, that represents whether or not the rocket is relaunchable), and LAB_ID (an integer representing what lab the rocket is being built in). LAB_ID is a foreign key that references a lab number in LAB.

Rocket_Composed_Of Table

```
CREATE TABLE Rocket_Composed_Of(
        ROCKET_ID INT NOT NULL,
        PART_ID INT NOT NULL,
        COMP_QUANT INT NOT NULL,
        PRIMARY KEY (ROCKET_ID, PART_ID),
        FOREIGN KEY (ROCKET_ID) REFERENCES ROCKET (ROCKET_ID),
        FOREIGN KEY (PART_ID) REFERENCES PART (PART_ID));
```

This creates the Rocket_Composed_Of entity. This table was created from the many-to-many relationship between ROCKET and PART. Its attributes consist of ROCKET_ID (an integer representing the rocket), PART_ID (an integer representing the part), and COMP_QUANT (an integer representing the number of each part that composes the rocket). The primary key is made up of ROCKET_ID and PART_ID, ROCKET_ID is a foreign key that references the rocket number in ROCKET, and PART_ID is a foreign key that references the part number in PART.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attribute COMP_QUANT functionally depends on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (there is only one non-prime attribute, can't functionally depend on another non-prime attribute).

Supplies_Equip

```
CREATE TABLE Supplies_Equip(
        MAN_ID INT NOT NULL,
        EQUIP_ID INT NOT NULL,
        SUPPLY_QUANT INT NOT NULL,
        SUPPLY_PRICE INT NOT NULL,
        PRIMARY KEY (MAN_ID, EQUIP_ID),
        FOREIGN KEY (MAN_ID) REFERENCES MANUFACTURER (MAN_ID),
        FOREIGN KEY (EQUIP_ID) REFERENCES EQUIPMENT (EQUIP_ID));
```

This creates the Supplies_Equip entity. This table was created from the many-to-many relationship between MANUFACTURER and EQUIPMENT. Its attributes consist of MAN_ID (an integer representing the manufacturer), EQUIP_ID (an integer representing the piece of equipment), SUPPLY_QUANT (an integer representing the amount of equipment that has been supplied), and SUPPLY_PRICE (an integer representing the price of each piece of equipment that was supplied). MAN_ID is a foreign key that references the manufacturer number in MANUFACTURER, and EQUIP_ID is a foreign key that references the equipment number in EQUIPMENT.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attributes SUPPLY_QUANT and SUPPLY_PRICE functionally depend on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (Neither SUPPLY_QUANT nor SUPPLY_PRICE are functionally dependent on each other.).

Supplies_Parts Table

```
CREATE TABLE Supplies_Parts(
        MAN_ID INT NOT NULL,
        PART_ID INT NOT NULL,
        SUPPLY_QUANT INT NOT NULL,
        SUPPLY_PRICE INT NOT NULL,
        PRIMARY KEY (MAN_ID, PART_ID),
        FOREIGN KEY (MAN_ID) REFERENCES MANUFACTURER (MAN_ID),
        FOREIGN KEY (PART_ID) REFERENCES PART (PART_ID));
```

This creates the Supplies_Parts entity. This table was created from the many-to-many relationship between MANUFACTURER and PART. Its attributes consist of MAN_ID (an integer representing the manufacturer), PART_ID (an integer representing the part), SUPPLY_QUANT (an integer representing the amount of each part that has been supplied), and SUPPLY_PRICE (an integer representing the price of each part that was supplied). MAN_ID is a foreign key that references the manufacturer number in MANUFACTURER, and PART_ID is a foreign key that references the part number in PART.

This table is in 1NF, because there are no repeating groups, and a primary key is defined such that each record is uniquely identifiable.
This table is in 2NF, because it is in 1NF, and there are no partial dependencies (the non-prime attributes SUPPLY_QUANT and SUPPLY_PRICE functionally depend on the entire composite key).
This table is in 3NF, because it is in 2NF, and there are no transitive dependencies (Neither SUPPLY_QUANT nor SUPPLY_PRICE are functionally dependent on each other.).

Queries

Query 1: All/every

List the first and last names of employees who work on the construction of every payload.

```
SELECT Employee.EMP_FNAME, Employee.EMP_LNAME
FROM Employee
WHERE NOT EXISTS
        (SELECT *
        FROM Payload
        WHERE NOT EXISTS
                (SELECT *
                FROM Emp_Constructs_Payload
                WHERE Employee.EMP_ID = Emp_Constructs_Payload.EMP_ID
                AND Emp_Constructs_Payload.PAYLOAD_ID = Payload.PAYLOAD_ID));
```

Query 2: Only

List research names that only has employees from the computing department working on it.

```
SELECT Research.RESEARCH_NAME
FROM Research
WHERE Research.RESEARCH_ID NOT IN
        (SELECT Emp_Conducts.RESEARCH_ID
        FROM Emp_Conducts
        WHERE Emp_Conducts.EMP_ID NOT IN
                (SELECT Employee.EMP_ID
                FROM Employee, Department
                WHERE Department.DEPART_ID = Employee.DEPART_ID
                AND Department.DEPART_NAME = ‘Computing’));
```

Query 3: None

List the first and last names of employees who hold no certifications.

```
SELECT Employee.EMP_FNAME, Employee.EMP_LNAME
FROM Employee
WHERE Employee.EMP_ID NOT IN
        (SELECT Emp_Cert.EMP_ID
        FROM Emp_Cert, Certification
        WHERE Emp_Cert.CERT_ID = Certification.CERT_ID);
```

Query 4: Left Join

List all medal names and years, and the full names of any employees who have earned it.

```
SELECT Medal.MEDAL_NAME, Medal.MEDAL_YEAR, Employee.EMP_FNAME,
Employee.EMP_LNAME
FROM Medal LEFT JOIN Emp_Award ON Medal.MEDAL_ID = Emp_Award.MEDAL_ID LEFT
JOIN Employee ON Emp_Award.EMP_ID = Employee.EMP_ID;
```

Query 5: Right Join

List the names of all parts and the names of any payloads that compose of them.

```
SELECT Part.PART_NAME, Payload.PAYLOAD_NAME
FROM Payload RIGHT JOIN Payload_Composed_Of ON Payload.PAYLOAD_ID =
Payload_Composed_Of.PAYLOAD_ID RIGHT JOIN Part ON Part.PART_ID =
Payload_Composed_Of.PART_ID
```

Query 6: Full Join

List the first and last names of every employee and the names of any certifications that they have, and list the names of every certification and the first and last names of any employee who have earned it.

```
SELECT Employee.EMP_FNAME, Employee.EMP_LNAME, Certification.CERT_NAME
FROM Certification FULL JOIN Emp_Cert ON Certification.CERT_ID = Emp_Cert.CERT_ID
FULL JOIN Employee ON Employee.EMP_ID = Emp_Cert.EMP_ID;
```

Query 7: 6 tables

List employees' first and last names who work in the engineering department who are certified spacecraft engineers, and the name of the payload they construct.

```
SELECT Employee.EMP_FNAME, Employee.EMP_LNAME, Payload.PAYLOAD_NAME
FROM Employee, Payload, Emp_Constructs_Payload, Department, Certification, Emp_Cert
WHERE Employee.EMP_ID = Emp_Cert.EMP_ID
AND Emp_Cert.CERT_ID = Certification.CERT_ID
AND Employee.EMP_ID = Emp_Constructs_Payload.EMP_ID
AND Emp_Constructs_Payload.PAYLOAD_ID = Payload.PAYLOAD_ID
AND Employee.DEPART_ID = Department.DEPART_ID
AND Department.DEPART_NAME = 'Engineering'
AND Certification.CERT_NAME = 'Certified Spacecraft Engineer';
```

Query 8

List missions names, the mission start and end dates, the first and last names of employees who operate them, the employee start date, their job titles, the salaries associated with the job titles, and the number of hours they have operated the mission, where the start date of the mission was before 1-Jan-2010 and the start date of the employee was before 1-Jun-1992.

```
SELECT Mission.MISSION_NAME, Mission.MISS_START_DATE, Mission.MISS_END_DATE,
Employee.EMP_FNAME, Employee.EMP_LNAME, Employee.EMP_START_DATE,
Job.JOB_TITLE, Job.SALARY, Emp_Operates.OPERATE_HRS
FROM Mission, Employee, Emp_Operates, Job
WHERE Mission.MISSION_ID = Emp_Operates.MISSION_ID
AND Emp_Operates.EMP_ID = Employee.EMP_ID
AND Employee.JOB_ID = Job.JOB_ID
AND Mission.MISS_START_DATE < ‘1-Jan-2010’
AND Employee.EMP_START_DATE < ‘1-Jun-1992’;
```

Query 9

Get the manufacturer name of manufacturers who have supplied a part that is supplied by another manufacturer who supplies a part that goes to constructing a payload which is being constructed in the satellite construction lab.

```
SELECT DISTINCT Manufacturer.MAN_NAME
FROM Manufacturer, Supplies_Parts SupPar1, Supplies_Parts SupPar2, Supplies_Parts
SupPar3, Part, Payload_Composed_Of, Payload, Lab
WHERE Manufacturer.MAN_ID = SupPar1.MAN_ID
AND SupPar1.PART_ID = SupPar2.PART_ID
AND SupPar1.MAN_ID <> SupPar2.MAN_ID
AND SupPar2.MAN_ID = SupPar3.MAN_ID
AND SupPar3.PART_ID = Part.PART_ID
AND Part.PART_ID = Payload_Composed_Of.PART_ID
AND Payload_Composed_Of.PAYLOAD_ID = Payload.PAYLOAD_ID
AND Payload.LAB_ID = Lab.LAB_ID
AND Lab.LAB_NAME = ‘Satellite Construction’;
```

Query 10

List the mission name, the launch id, and the rocket name of missions that have involved a launch with a relaunchable rocket that has been constructed only by employees from the engineering department.

```
SELECT Mission.MISSION_NAME, Launch.LAUNCH_ID, Rocket.ROCKET_NAME
FROM Mission, Launch, Rocket
WHERE Mission.MISSION_ID = Launch.MISSION_ID
AND Launch.ROCKET_ID = Rocket.ROCKET_ID
AND Rocket.RELAUNCHABLE = 'Y'
AND Rocket.ROCKET_ID NOT IN
        (SELECT Emp_Constructs_Rocket.ROCKET_ID
        FROM Emp_Constructs_Rocket
        WHERE Emp_Constructs_Rocket.EMP_ID NOT IN
                (SELECT Employee.EMP_ID
                FROM Employee, Department
                WHERE Employee.DEPART_ID = Department.DEPART_ID
                AND Department.DEPART_ID = 'Engineering');
```